Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, June 14-19, 2009

A Modified Sparse Distributed Memory Model for Extracting Clean Patterns from Noisy Inputs

Hongying Meng[a] Kofi Appiah[a] Andrew Hunter[a] Shigang Yue[a]
Mervyn Hobden[b] Nigel Priestley[b] Peter Hobden[b] Cy Pettit[b]

Abstract— **The Sparse Distributed Memory (SDM) proposed by Kanerva provides a simple model for human long-term memory, with a strong underlying mathematical theory. However, there are problematic features in the original SDM model that affect its efficiency and performance in real world applications and for hardware implementation. In this paper, we propose modifications to the SDM model that improve its efficiency and performance in pattern recall. First, the address matrix is built using training samples rather than random binary sequences. This improves the recall performance significantly. Second, the content matrix is modified using a simple tri-state logic rule. This reduces the storage requirements of the SDM and simplifies the implementation logic, making it suitable for hardware implementation. The modified model has been tested using pattern recall experiments. It is found that the modified model can recall clean patterns very well from noisy inputs.**

I. Introduction

In the human brain, the long-term memory can store large quantities of information for very long durations (sometimes a whole life span). For example, we can remember telephone numbers for many years through repetition-training; this information is said to be stored in long-term memory. It has been found that long-term memory encodes information semantically[1]. The memory is also associative [2], recalling data when an input pattern is sufficiently close to the stored pattern. There is therefore a long-standing research interest in associative memory models.

The Hopfield neural-network model [3] is attractive for its simplicity and its ability to function as a massively parallel, autoassociative memory. However, it is not suitable for human memory as it is quite limited in its ability to store sets of correlated patterns [4].

Kanerva's Sparse Distributed Memory (SDM) [5][6] was developed as an abstract mathematical model of human long-term memory. It is a simple content-addressable memory, with some architectural similarity to the structure of the cerebellum, and is able to store randomly distributed input data quite effectively. However, its efficiency in handling non random data, is poor. In order to improve its performance, the SDM model has been treated and modified as a neural network [7][8][9][10][11].

Hely *et. al.* [12] introduced an alternative SDM, the SDM signal model, which retains the essential characteristics of the original SDM, whilst providing the memory with a greater scope for plasticity and self-evolution, including by training the address layer. By removing many of the problematic features of the original SDM, this model is not as dependent upon *a priori* input values. This gives it an increased robustness to learn either random or correlated input patterns. This new SDM signal model outperforms the previous modified single layer neural network [7]. The most significant modification in [12] is that the address matrix is modified by training samples instead of random initialization.

The SDM model has been successfully applied in many applications such as number or letter recognition [13], and handwriting character recognition [14]. It may also be implemented in hardware [15][16] because of its parallel architecture and simple writing and reading operations.

Recently, the capability of storing and recalling patterns containing rank-order information has been studied. It has been found that the modified SDM model could efficiently store and recover rank-ordered codes [17][18][19][20].

In this paper, we try to use the SDM model to extract some patterns from the sample dataset. In comparison with other memory model or data retrieval methods, the pattern or feature recalled by the SDM model is not exactly same as one of the input data. Based on the idea of [12] on using training samples in address matrix, we introduce a more direct method to create the address matrix from training data. We also introduce the tri-state logic rule in the content matrix, which reduces the storage size of the matrix and simplifies the learning rule. The new model is consequently much more suitable for efficient hardware implementation.

The rest of this paper is organized as follows: Section II introduced Kanerva's SDM model, which forms the base for the modified model. Section III introduces the modified SDM model where we use several methods to improve its performance. Some simulation results are presented in Section IV. Section V concludes the paper.

II. SDM Model

Kanerva's SDM model is illustrated in figure 1. In this model, there are two main matrices: the *address matrix*, A, and the *content matrix*, C. In the original SDM model, matrix A is a collection of random (possibly sparse) binary vectors. The Hamming distance is used to compute the difference between input addresses and hard locations. Hard locations with small differences to input addresses are called *active locations*. The values (integers) in the content matrix C of the active locations are incremented or decremented based on the values of the input data sequence.

[a] Department of Computing and Informatics, University of Lincoln, UK (corresponding email: hmeng@lincoln.ac.uk)
[b] E2V Technologies PLC, Lincoln, UK.

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

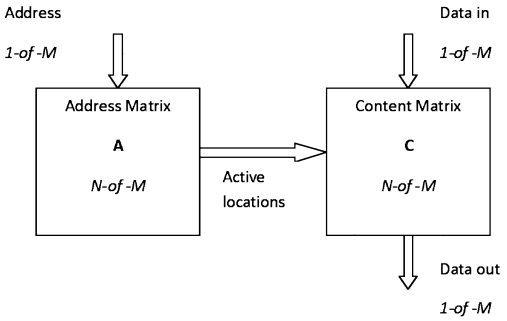

Fig. 1. Kanerva's SDM model

The SDM model can be described algebraically. Assume the binary pattern/feature has dimension of M, then the pattern space will be $\{0,1\}^M$. Then the address matrix A can be represented as the equation 1.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \cdots & & & \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_N \end{bmatrix} \quad (1)$$

where $a_{ij} \in \{0,1\}\,(i = 1,2,\cdots,N, j = 1,2,\cdots,M)$ are binary numbers. Therefore, matrix A is a binary matrix with size of $N \times M$.

The content matrix C in the SDM model can be represented using the equation 2.

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1M} \\ c_{21} & c_{22} & \cdots & c_{2M} \\ \cdots & & & \\ c_{N1} & c_{N2} & \cdots & c_{NM} \end{bmatrix} \quad (2)$$

where $c_{ij}\,(i = 1,2,\cdots,N, j = 1,2,\cdots,M)$ are integers. Therefore, matrix C has size $N \times M$. In addition to these two matrices, there are some control parameters in the model.

The SDM model has several operational steps.

A. Initialization

The two matrices are initialized as follows: C is zeroed, and A is randomly initialized. The ratio of number of 1 compared with number of 0 during the initialization of A is denoted as ρ (typically set to 0.1 or 0.5). This random initialization of the address matrix A works very well if the stored data is random, or at least fairly uniformly distributed across the input space. However, the performance on non-random data is very poor, as only a small part of the address matrix actually get used.

B. Writing Operation

Like a computer memory, the writing operations save data in the memory model. The SDM model modifies its content to store this information. There are two input binary vectors for writing: address vector $x = (x_1, x_2, \cdots, x_M)$ and data vector $d = (d_1, d_2, \cdots, d_M)$.

For the given input address $x = (x_1, x_2, \cdots, x_M)$, the Hamming distances $H(x, a_i)$ between x and every row a_i of matrix A is calculated and thresholded. A new vector $s = (s_1, s_2, \cdots, s_M)$ is used to label the active locations within the threshold distance.

$$s_i = \begin{cases} 1 & H(x,a_i) < T \\ 0 & H(x,a_i) \geq T \end{cases} \quad (3)$$

The Hamming distance of two binary sequences $p = (p_1, p_2, \cdots, p_M)$ and $q = (q_1, q_2, \cdots, q_M)$ is calculated in equation 4:

$$H(p,q) = \sum_{i=1}^{M} I(p_i, q_i) \quad (4)$$

where I is defined as equation 5:

$$I(p_i, q_i) = \begin{cases} 1, & p_i \neq q_i \\ 0, & p_i = q_i \end{cases} \quad (5)$$

The content of the matrix C is updated for the active locations based on the results of vector s in equation 6. A '1' in the input data d increases by 1 the values in the corresponding locations and a 0 decreases the value of the counter in the corresponding locations.

$$c_{i,j} = \begin{cases} c_{i,j} + 1 & s_i = 1, d_j = 1 \\ c_{i,j} - 1 & s_i = 1, d_j = 0 \\ c_{i,j} & s_i = 0 \end{cases} \quad (6)$$

C. Reading Operation

To read a datum according to an input address x, the memory works similarly to the writing operation except that input data register d is not used.

For a given input address $x = (x_1, x_2, \cdots, x_M)$, the Hamming distances $H(x, a_i)$ between x and every row a_i of matrix A is calculated and thresholded. The vector s is defined based on equation 3. Then, a new vector h in equation 7 is created by summing the related elements in the content matrix C. These sums are then thresholded at zero, which generates a 1 in the j^{th} bit if the j^{th} sum is greater than or equal to zero, and a 0 if the sum is smaller than zero. The thresholded value will be the output data in the output register o. That is,

$$h_j = \sum_{i=1}^{N} s_i \times c_{ij} \quad (7)$$

$$o_j = \begin{cases} 1 & h_j > 0 \\ 0 & h_j \leq 0 \end{cases} \quad (8)$$

III. Modified SDM model

A number of modifications to the original SDM model have been proposed, to address limitations of the model. One significant issue is the address matrix initialization. The performance of the original SDM model is very poor for non-random data. Some methods have been proposed to deal with this problem such as using a genetic algorithm [21]. Another disadvantage of the SDM model in comparison with RAM-based neural networks [22] is that the content matrix C is integer valued; this makes it relatively expensive to produce hardware implementations of SDM.

A. Sample-addressed address matrix A

Hely *et al.* [12] introduced the use of sample data in the address matrix; a similar idea was proposed in [8]. This greatly improves the performance on non-random input data. A randomly populated address matrix is able to memorize inputs across the whole address space. However, this space is typically very large, and in most applications only small regions are actually populated with data. We therefore require a very large number of hard locations to support reasonably fine discrimination between patterns.

We use some training samples to initialize the address matrix in a very straightforward fashion, copying the training samples $x = (x_1, x_2, \cdots, x_M)$ directly into the hard locations a_j. For each training sample, we first check whether the exact same pattern is already stored in the hard locations, and if so discard it (so that there are no duplicate rows in the trained matrix A).

B. Hard binary for the content matrix

One of the most costly elements in the original SDM model is the use of integer counters in the context matrix. These require both relatively high storage and expensive increment/decrement logic, when considered for parallel hardware implementation. We have therefore simplified the operation to use a more compact and cheaply implemented approach.

The first approach we tried was to use a binary content matrix C. The data vector d was copied into the active locations of the content matrix during training. The initial content matrix is empty. When a training sample is input, we calculate the Hamming distance between the input and the sample-addressed address matrix A. For the active locations, we replace the content in the active row by the input data binary vector. In this case, during the writing operation, instead of using equation 6, we use the following equation 9.

$$c_{i,j} = \begin{cases} 1 & s_i = 1, d_j = 1 \\ 0 & s_i = 1, d_j = 0 \\ c_{i,j} & s_i = 0 \end{cases} \tag{9}$$

In the reading operation, the threshold is set slightly higher because the minimum value in content matrix C is 0. In this case, the storage for the content matrix C is at least 8 times smaller than the original SDM model.

C. Tri-state logic

The use of "hard" binary in the content matrix C greatly reduces the storage requirements, but also discards the counting information available from the training samples. To capture some of this information, but retaining a highly efficient structure, we have introduced the use of a tri-state rule.

In digital electronics three-state, tri-state, or 3-state logic allows output ports to have a value of logical 0, 1, or Hi-Z. A Hi-Z output puts the pin in a high impedance state, effectively removing the pin from its influence on the circuit.

In binary logic the two levels are logical high and logical low, which generally correspond to a binary 1 and 0 respectively. Signals with one of these two levels can be used in boolean logic for digital circuit design or analysis. In three-state logic, an output device can also be high impedance. This is not a logic level, but means that the output does not control the state of the connected circuit.

The tri-state logic between two bits u and v can be represented in the equation 10

$$T_{u,v} = \begin{cases} 1, & u = v = 1 \\ 0, & u = v = 0 \\ Z, & otherwise \end{cases} \tag{10}$$

Like the hard binary method in the previous subsection, we change the equation 6 in the writing operation of the SDM model. We use the following equation 11 to update the content matrix C.

$$c_{i,j} = \begin{cases} 1 & s_i = 1, d_j = 1, c_{i,j} = 1 \\ 1 & s_i = 1, d_j = 1, c_{i,j} = Z \\ 0 & s_i = 1, d_j = 0, c_{i,j} = 0 \\ 0 & s_i = 1, d_j = 0, c_{i,j} = Z \\ Z & s_i = 1, d_j = 0, c_{i,j} = 1 \\ Z & s_i = 1, d_j = 1, c_{i,j} = 0 \\ c_{i,j} & s_i = 0 \end{cases} \tag{11}$$

During the read operation, value Z is ignored in the sum in equation 7. For the storage of content matrix C, every element only occupies 2 bits. It is somewhat smaller than an integer counter, that might typically use 8 or 16 bits, and is much more suitable for parallel implementation as the logic operations involved are very simple.

IV. Experimental Results

In order to compare the efficiency and performance of our Tristate SDM model with the original SDM model, we create two simple pattern datasets. Both of these data sets were created based on clean patterns, but all the training and testing samples have some added noise. Consequently, the inputs of the SDM model are always noisy patterns. In order to keep the experiment simple, for every input sample we let the input address vector be exactly the same as the input data vector.

A. Experiments on dataset 1

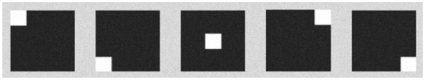

Fig. 2. The 5 clean patterns in dataset 1

The first dataset is based on the five patterns shown in figure 2. Each sample is an 8×8 image. Then, we add some noise into every samples of the patterns. The values for each pixel are 0 or 1. We give a probability of 5 percent chance to change its valuse on each pixel. Because there are 64 pixels in one sample, about 3 or 4 pixels are changed values from

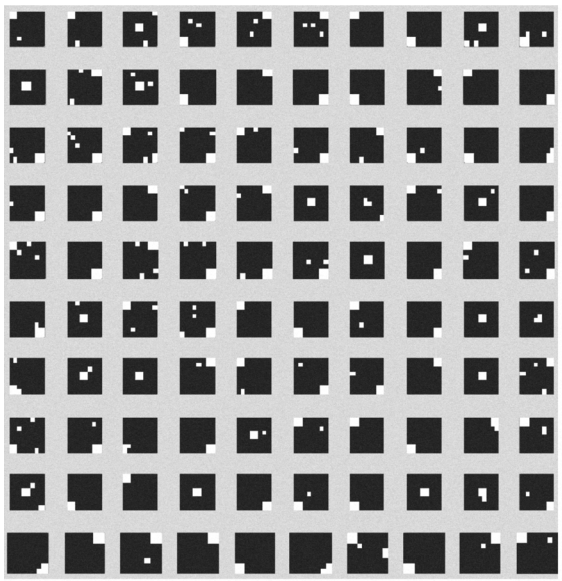

Fig. 3. 100 testing samples in dataset 1. Image size is 8×8 with 5% noise added.

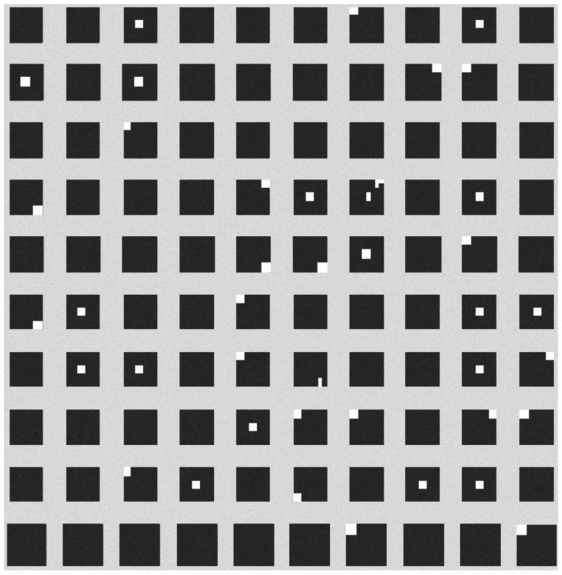

Fig. 4. The recall output of the the original SDM model. The address matrix is initialized randomly with $\rho = 0.5$. The content matrix is integer valued.

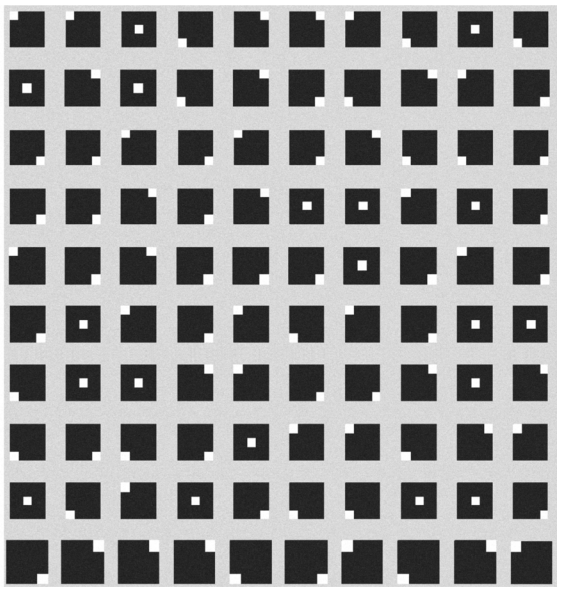

Fig. 5. The recall output of the sample-addressed SDM model. The address matrix contains training samples; the content matrix is integer valued.

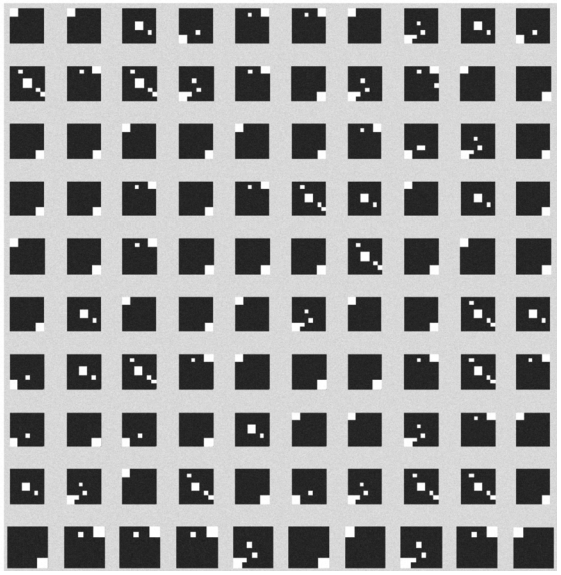

Fig. 6. The recall output of the Binary SDM model. The address matrix contains training samples; the hard binary rule is applied to the content matrix.

0 to 1 or from 1 to 0. Figure 3 shows 100 testing examples. On each image, white pixel stands for value 1 and black pixel stands for value 0. In the following experiment, the dimension of the binary pattern is $M = 64$ and the number of hard locations is $N = 4000$.

These experiments clearly show several facts. First, the original SDM model has some limitations in recalling the non-random patterns correctly (we experimented with a number of different thresholds, and figure 4 represents the best results achieved). Second, figure 5 shows that we can extract clean patterns from noisy inputs if we fill the address matrix with training samples. Third, figure 7 showed that we can get similar results if we use tri-state rules in the content matrix C. In this case, only 2 bits are needed to store the elements of the content matrix, and the addressing logic is simple. Finally, figure 6 showed that we lose some performance if we further reduce the storage of the content matrix into a purely binary matrix.

B. Experiments on dataset 2

In dataset 2, the pattern is more complex than that in dataset 1. The four patterns are 'rectangle', 'cross', 'plus' and 'circle'. The size of the image are still 8×8, so the dimension of the binary pattern is $M = 64$. We kept the

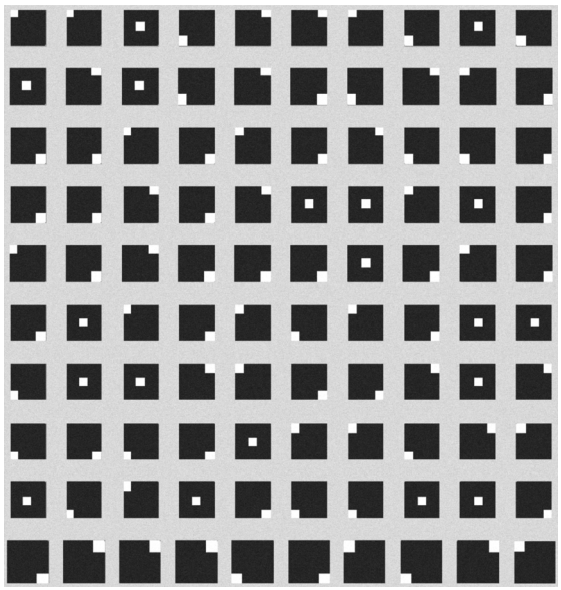

Fig. 7. The recall output of the Tristate SDM model. The address matrix contains training samples. The tri-state rule is applied to the content matrix.

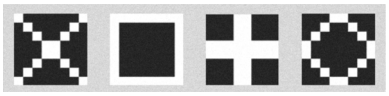

Fig. 8. The 4 clean patterns in dataset 2

number of hard location at $N = 4000$, and added 5% noise to every sample as that for dataset 1. Figure 8 shows the 4 clean patterns. Figure 9 shows 100 testing examples.

The experimental results are shown in figures 10, 11, 12 and 13 respectively. From these figures, we can clearly see that the results on dataset 2 are quite similar to those on dataset 1. The results in figure 13 are not as good as those obtained by the integer-valued SDM model using a trained address matrix, but are still impressive. The correct patterns are retrieved, but the exemplar reconstruction is imperfect.

V. Conclusion

This paper proposes a modified SDM model with several changes to the original SDM model. First, the address matrix has been initialized by training samples in a straightforward fashion. This significantly improves the recall performance. Second, two different methods have been used to implement the content matrix, in order to save storage for the hardware implementation. The tri-state technique reduces the storage requirement significantly while keeping relatively high performance. The new model is ideal for hardware implementation of a sparse distributed memory. It also can be treated as a pattern extraction machine to extract clean patterns automatically from noisy inputs. We are currently working on experiments to mix different type of patterns together in the training and testing. Our future work will focus on the FPGA implementation of a powerful and parallel SDM model as a pattern extraction and recognition machine.

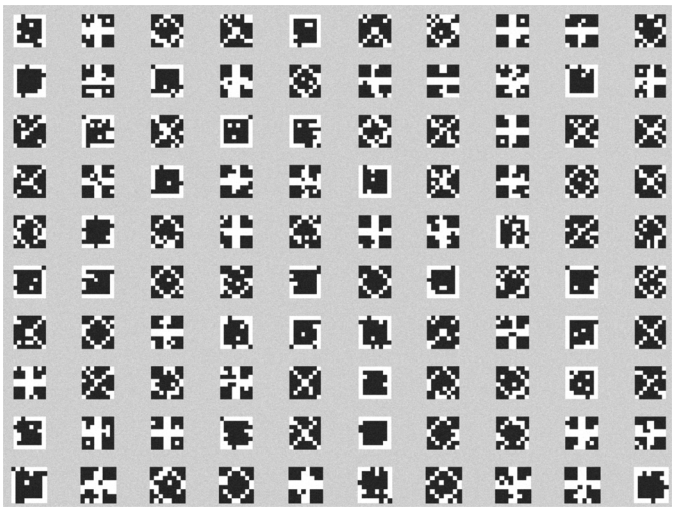

Fig. 9. 100 testing samples in dataset 2

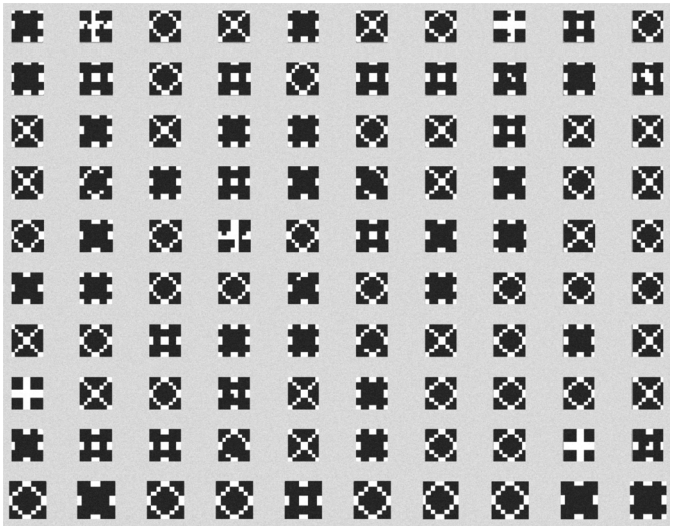

Fig. 10. The recall output of the original SDM model. The address matrix is initialized randomly with $\rho = 0.5$. The content matrix is integer valued.

References

[1] A. D. Baddeley, "The influence of acoustic and semantic similarity on long-term memory for word sequences." *Quart. J. exp. Psychol.*, vol. 18, pp. 302–309, 1966.

[2] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-holographic associative memory." *Nature*, vol. 222, pp. 960–962, 1969.

[3] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, 1982.

[4] J. D. Keeler, "Comparison between kanerva's sdm and hopfield-type neural networks," *Cognitive Science*, vol. 12, no. 3, pp. 299–329, 1988.

[5] P. Kanerva, *Sparse distributed memory*. The MIT Press, 1988.

[6] ——, "Sparse distributed memory and related models," in *Associative Neural Memories: Theory and Implementation*, 1993, pp. 50–76.

[7] R. W. Prager and F. Fallside, "The modified kanerva model for automatic speech recognition." *Computer Speech and Language*, vol. 3, p. 6181, 1989.

[8] R. Prager, "Some experiments with fixed non-linear mappings and single layer networks." 1992, technical report, Cambridge University, Engineering Department CUED/F-INFENG/TR.106.

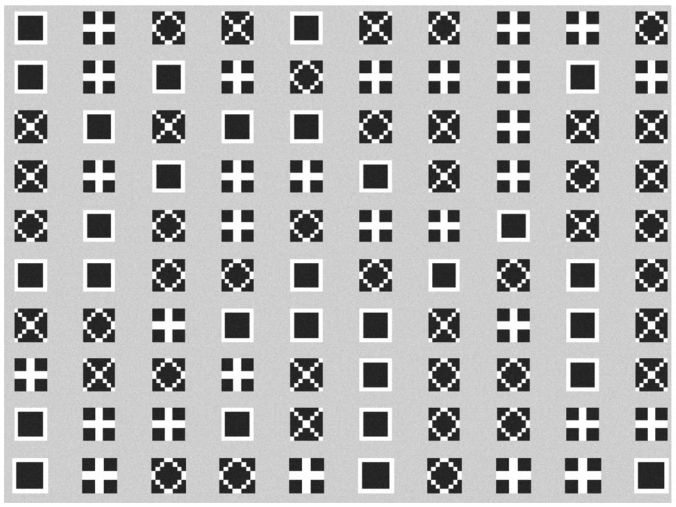

Fig. 11. The recall output of the sample-addressed SDM model. The address matrix is filled using training samples; the content matrix is integer valued.

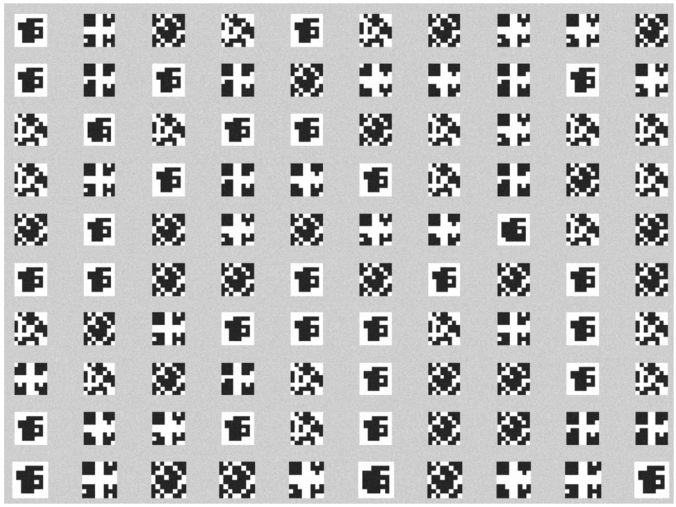

Fig. 12. The recall output of the Binary SDM model. The address matrix is filled by samples; the content matrix contains binary patterns.

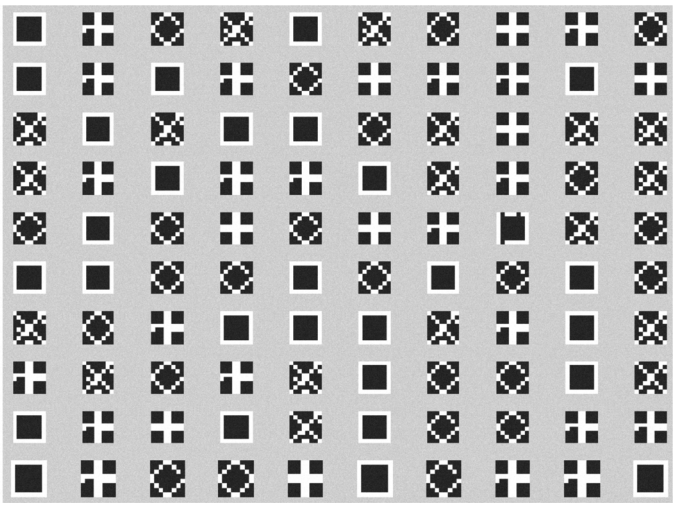

Fig. 13. The recall output of the Tristate SDM model. The address matrix is filled with training samples; the tri-state rule is applied to the content matrix.

[9] Y. Baram, "Memorizing binary vector sequences by a sparsely encoded network." *IEEE Transactions on Neural Networks*, vol. 5, pp. 974–981, 1994.

[10] S. Ryan and J. Andreae, "Improving the performance of kanerva's associate memory." *IEEE Transactions on Neural Networks*, vol. 6, pp. 125–130, 1995.

[11] S. Holden and P. Rayner, "Generalization and pac learning: some new results for the class ofgeneralized single-layer networks." *IEEE Transactions on Neural Networks*, vol. 6, pp. 368–380, 1995.

[12] T. A. Hely, D. J. Willshaw, and G. M. Hayes, "A new approach to kanerva's sparse distributed memory," in *In IEEE Transactions on Neural Networks*, 1997, pp. 101–105.

[13] F. Zboril, "An application of the sparse distributed memory," in *In: Proceedings of the ASIS*, 1997, p. p.

[14] K. Fan and Y. Wang, "A genetic sparse distributed memory approach to the application of handwritten character recognition," vol. 30, no. 12, pp. 2015–2022, December 1997.

[15] M. Lindell, J. Sarrinen, J. Tomberg, P. Kanerva, and K. Kaski, "Configurable sparse distributed memory hardware implementation." in *IEEE International Sympoisum on Circuits and Systems*, 1991, pp. 3078–3081.

[16] A. Turk, A. Turk, G. Gorz, and G. Gorz, "Kanerva's sparse distributed memory: Object-oriented implementation on the connection machine," in *In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995.

[17] L. Perrinet, M. Samuelides, and S. Thorpe, "Coding static natural images using spiking event times: do neurons cooperate?" *IEEE Transactions on Neural Networks*, vol. 15, pp. 1164–1175, 2004.

[18] S. B. Furber, W. J. Bainbridge, J. M. Cumpstey, and S. Temple, "Sparse distributed memory using n-of-m codes," *Neural Netw.*, vol. 17, no. 10, pp. 1437–1451, 2004.

[19] M. Rehna and F. T. Sommer, "Storing and restoring visual input with collaborative rank coding and associative memory." *Neurocomputing*, vol. 69, pp. 1219–1223, 2006.

[20] S. B. Furber, G. Brown, J. Bose, J. M. Cumpstey, P. Marshall, and J. L. Shapiro, "Sparse distributed memory using rank-order neural codes." *IEEE Transactions on Neural Networks*, vol. 18, pp. 648–659, 2007.

[21] A. Anwar, D. Dasgupta, and S. Franklin, "Using genetic algorithms for sparse distributed memoryinitialization," in *Proceedings of the 1999 Congress on Evolutionary Computation*, 1999.

[22] J. Austin, *RAM-based neural networks*. World Scientific Publishing Co., 1998.